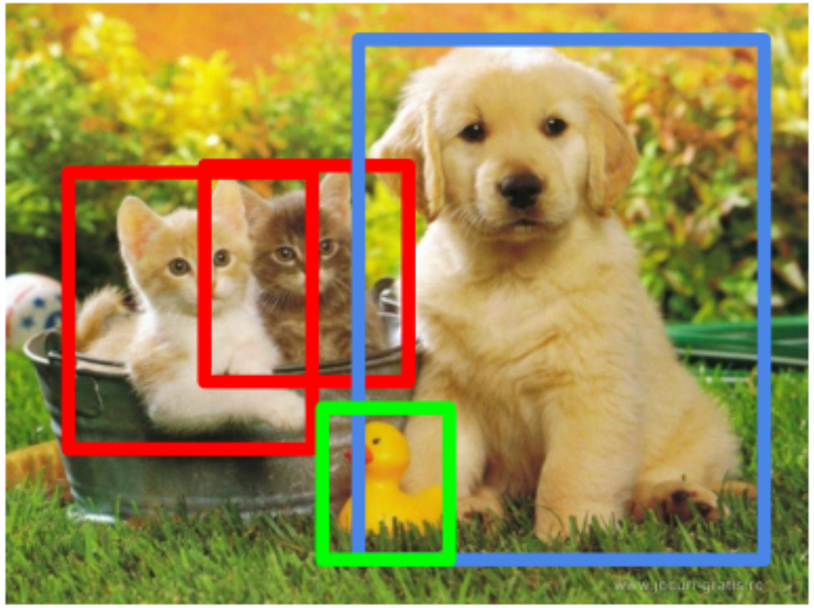
Dynamic R-CNN: Towards High Quality Object Detection via Dynamic Training

Hongkai Zhang[1,2], Hong Chang[1,2], Bingpeng Ma[2], Naiyan Wang[3], Xilin Chen[1,2]

[1] Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS)
[2] University of Chinese Academy of Sciences (UCAS), [3] TuSimple

Object Detection

- Goal
  - Localize all the objects in an image
  - Decide semantic categories of the objects



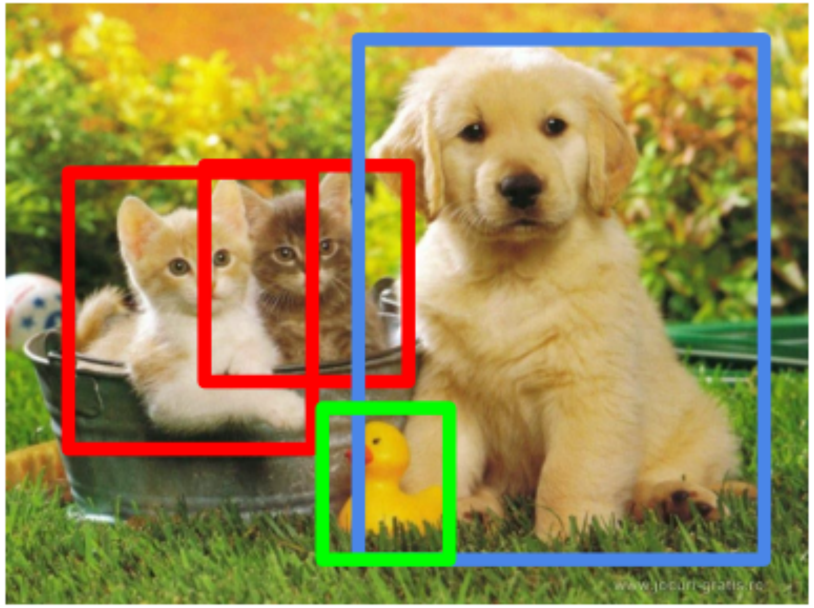
CAT, DOG, DUCK

Image source: CS231n Lecture, Stanford University.

Object detection aims to localize the objects in an image and decide their semantic categories.

Object Detection

- Two major categories
  - One-stage Detectors
  - Two-stage Detectors



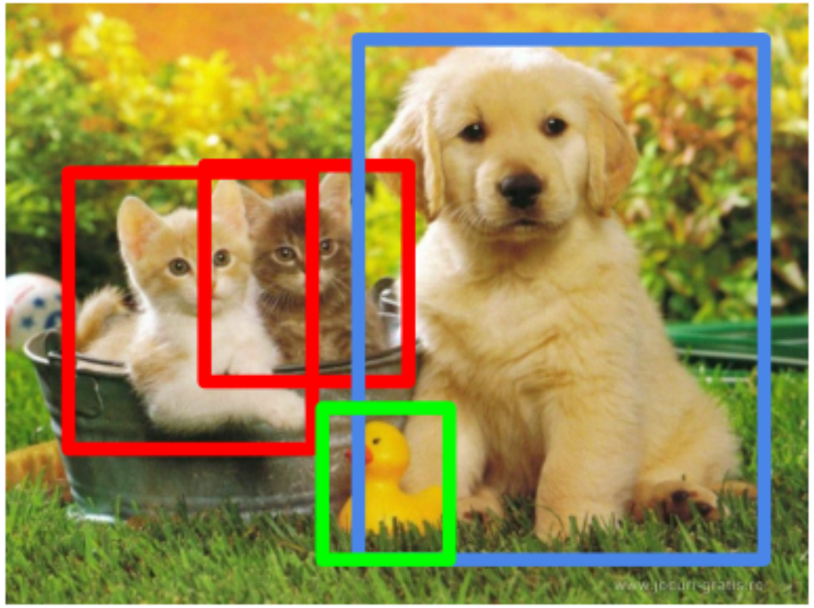
CAT, DOG, DUCK

Image source: CS231n Lecture, Stanford University.

Modern detection frameworks can be divided into two categories of one-stage detectors and two-stage detectors.

Object Detection

- Two major categories
  - One-stage Detectors
  - **Two-stage Detectors**



CAT, DOG, DUCK

Image source: CS231n Lecture, Stanford University.

In this work, we focus on the second one.

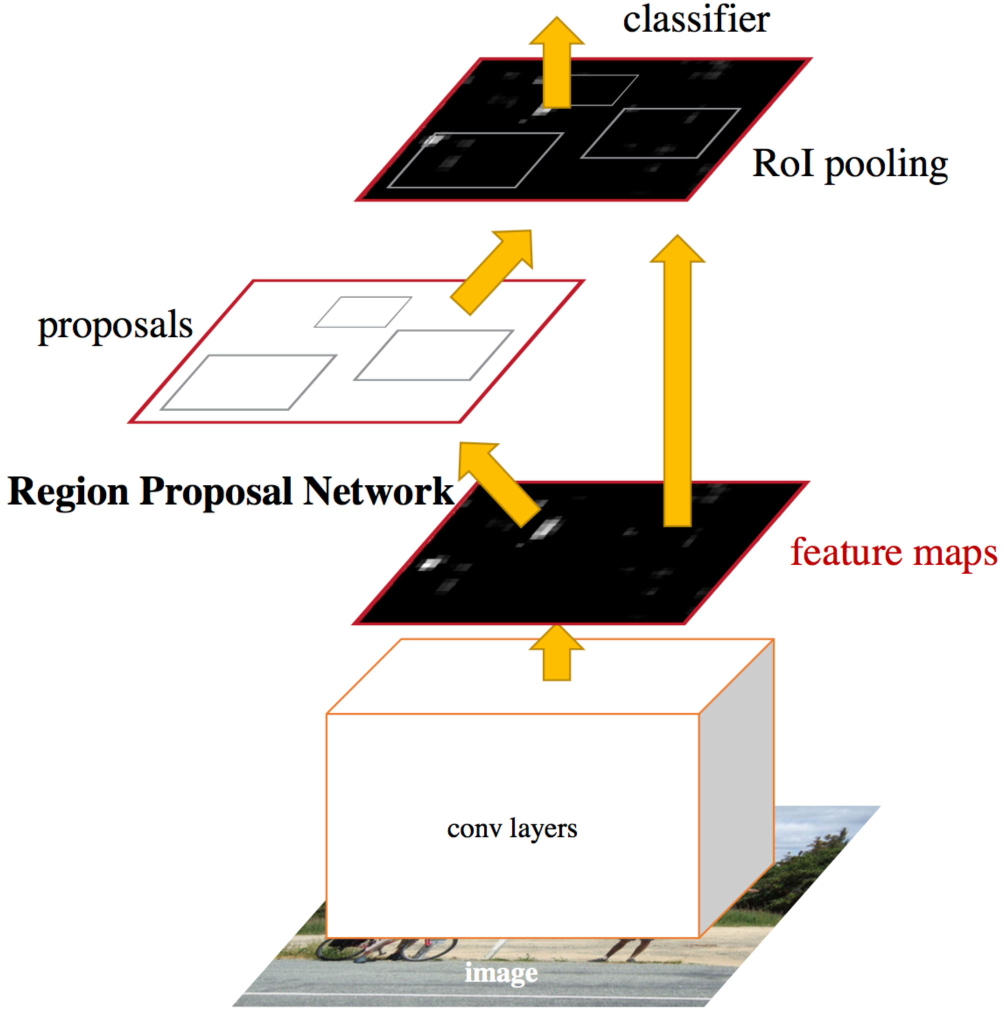
High Quality Object Detection

- Our goal: High Quality Object Detection
- What is ”high quality”?
  - Generally speaking, it stands for the results under high IoU
- Why this goal?
  - The higher the accuracy of the result -> the better

Our goal is to achieve high quality object detection, which requires more accurate results especially under high IoU.

Faster R-CNN [1]

- Two-stage detectors

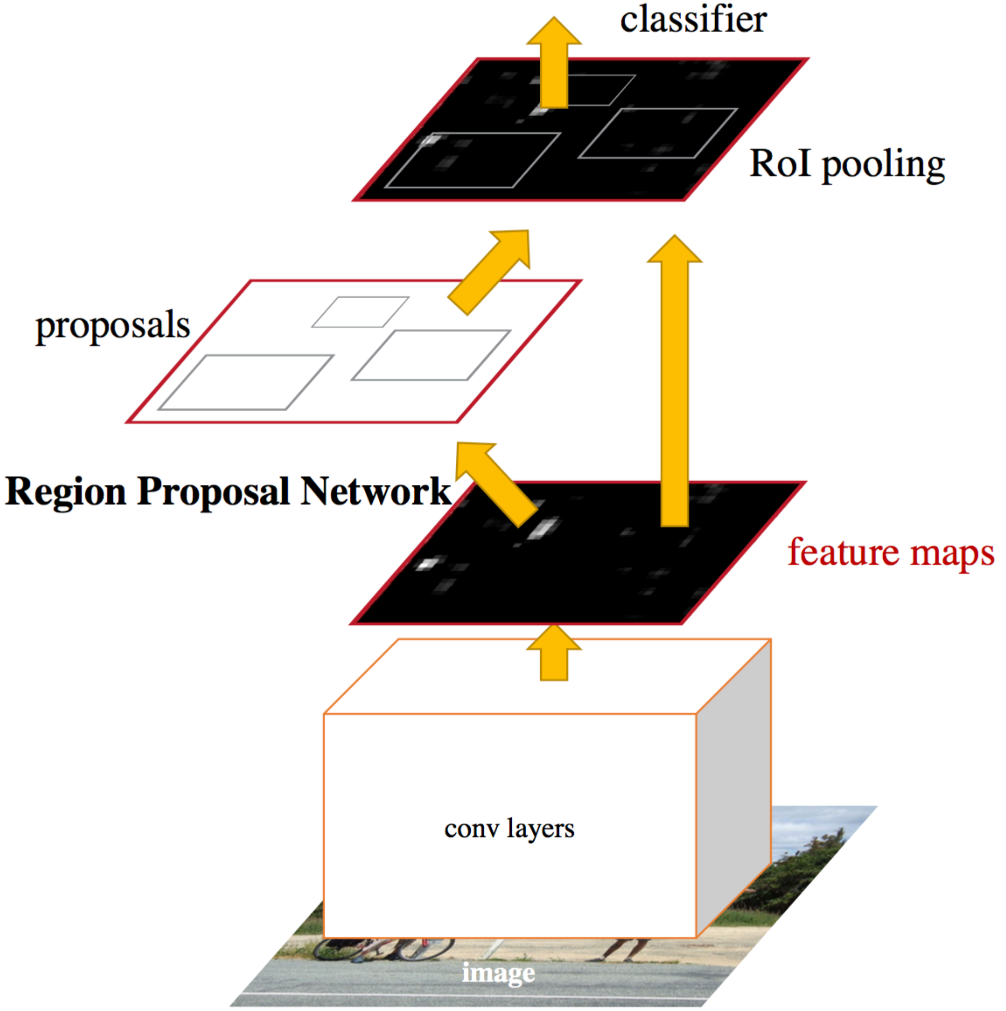


[1] Shaoqing Ren, et al. Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS 2015.

To better understand our work, let's first recall the structure of a representative two-stage detector Faster R-CNN.

Faster R-CNN [1]

- Two-stage detectors
  - Coarse-to-fine manner
    - Stage 1: Anchor -> Proposal
    - Stage 2: Proposal -> Prediction



[1] Shaoqing Ren, et al. Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS 2015.

Faster R-CNN follows a coarse-to-fine manner which first refines anchors to get the region proposals, then refines proposals and get the final predictions.

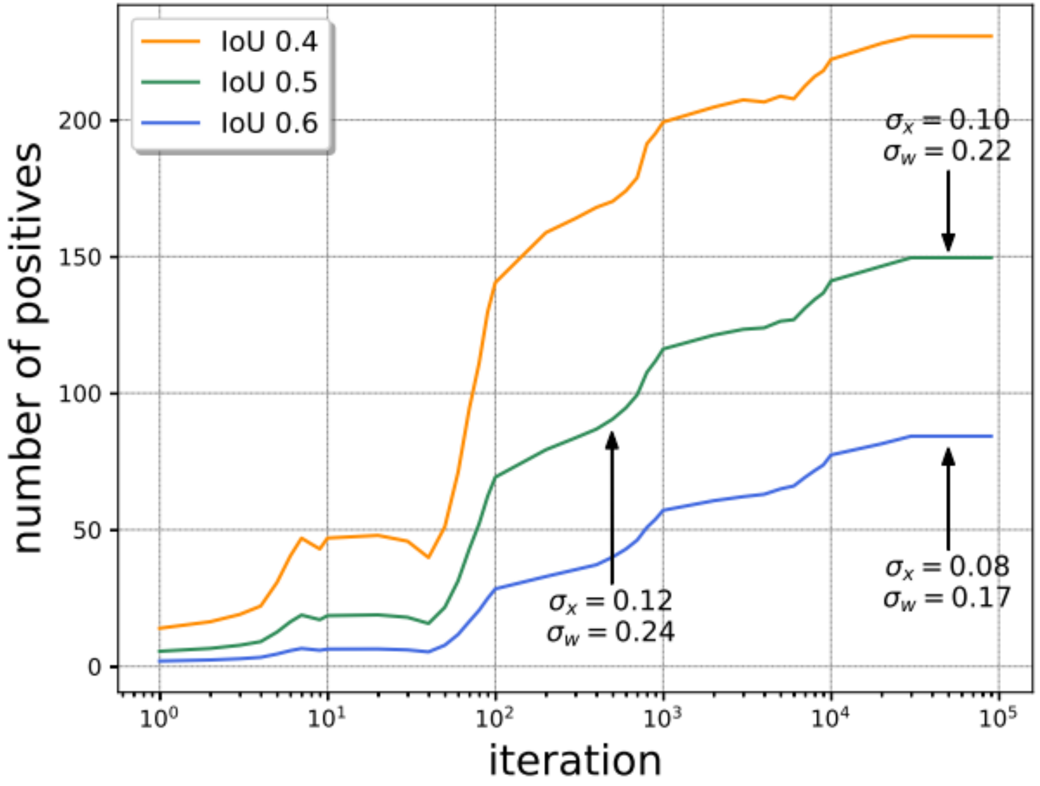
Inconsistency Problem in Faster R-CNN

- The dynamic training and fixed settings are inconsistent

We find that there exist inconsistency problems in Faster R-CNN, especially between the fixed network settings and the dynamic property in training.

Inconsistency Problem in Faster R-CNN

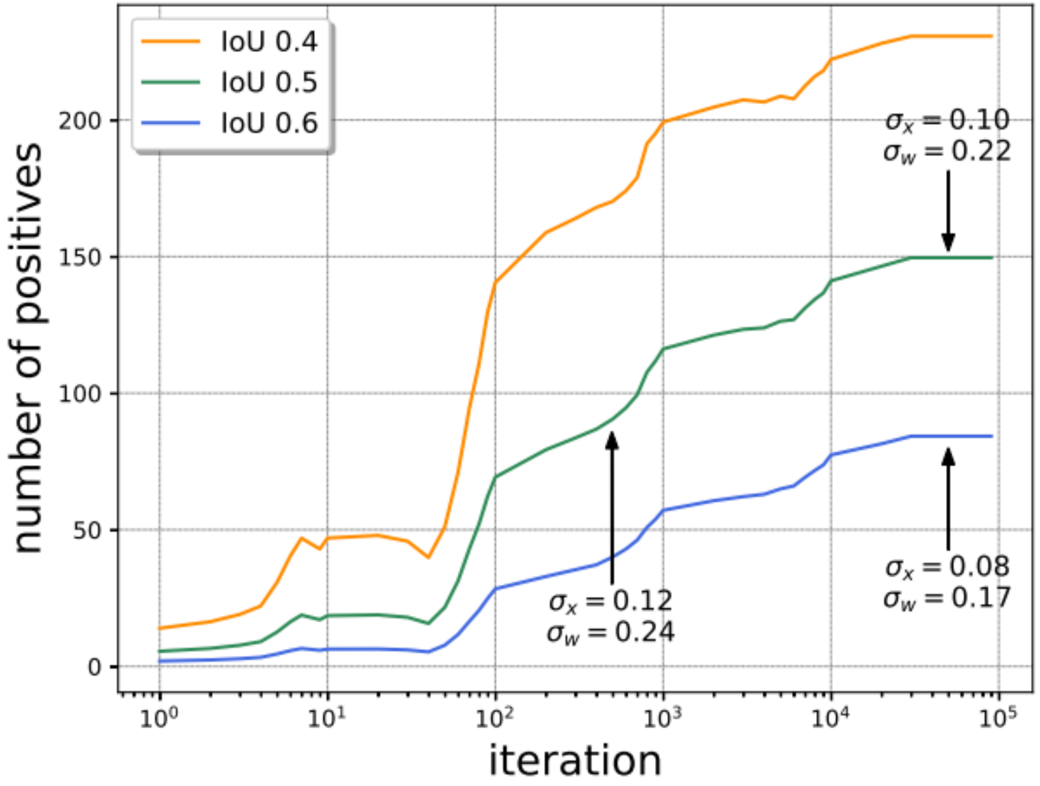
- The dynamic training and fixed settings are inconsistent
  - Dynamic: R-CNN input is the proposals



Specifically, the input of R-CNN is the proposals. As the training goes, the quality of proposals actually improves regardless of the certain IoU.

Inconsistency Problem in Faster R-CNN

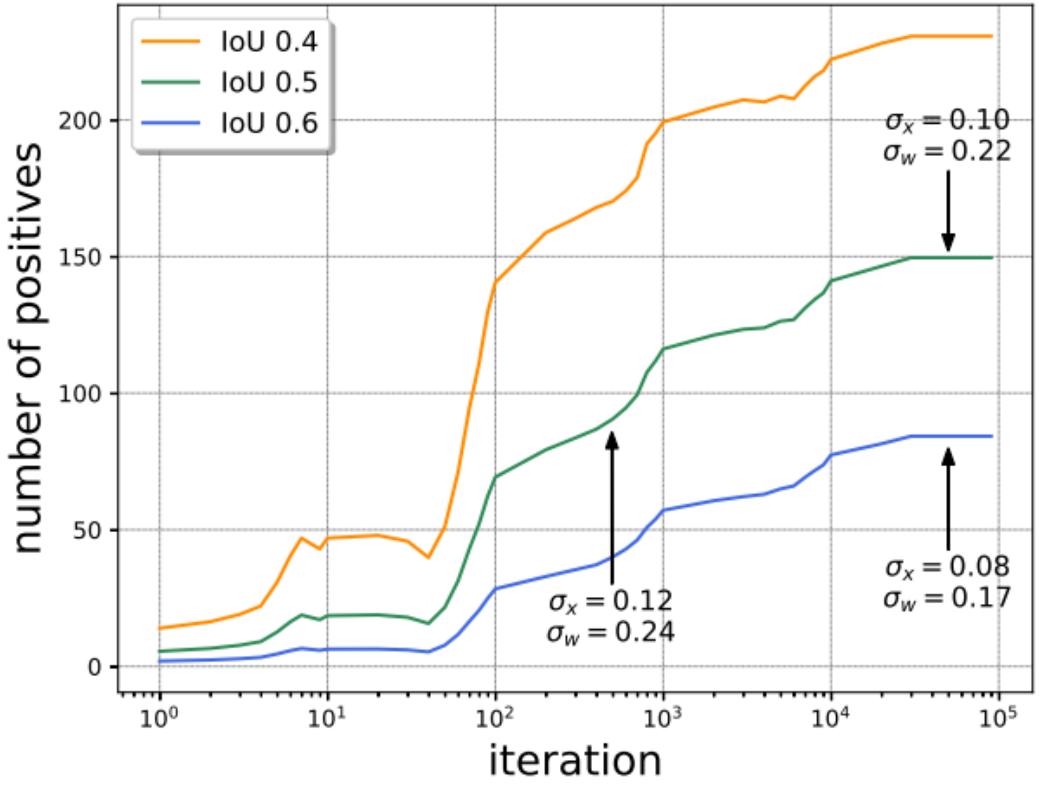
- The dynamic training and fixed settings are inconsistent
  - Dynamic: R-CNN input is the proposals
  - Fixed: network settings
- Harmful to high quality Object Detection



However, most of the network settings of the detection systems are fixed during training, which is harmful to achieving high quality object detection.

Inconsistency Problem in Faster R-CNN

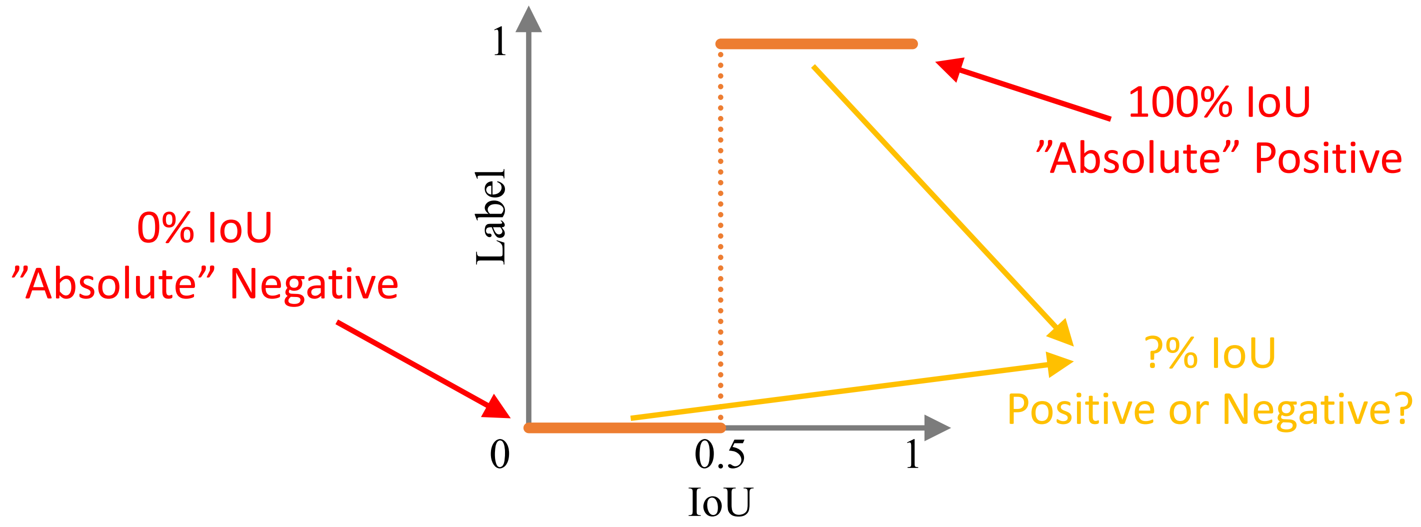
- The dynamic training and fixed settings are inconsistent
  - Dynamic: R-CNN input is the proposals
  - Fixed: network settings
- Harmful to high quality Object Detection
  - Proposal classification
  - Bounding box regression



For better understanding, we will discuss the effects of the inconsistency problem from two subtasks: proposal classification and bounding box regression.

Proposal Classification

- How to assign labels is an interesting question
  - Assignment strategy in Faster R-CNN (T+ = T- = 0.5)



  - Why the threshold is fixed at 0.5?

For proposal classification, it is not clear how to assign positive and negative labels for the proposals since the separation may be ambiguous.

Proposal Classification

- Training with different IoU thresholds will lead to classifiers with corresponding quality [2]

Backbone	IoU	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
ResNet-50-FPN	0.4	35.4	**58.2**	53.0	44.1	29.2	7.3
ResNet-50-FPN	0.5	36.6	58.1	**53.5**	**45.8**	31.5	8.8
ResNet-50-FPN	0.6	35.7	56.0	51.6	44.5	**31.6**	**9.3**

[2] Zhaowei Cai, et al. Cascade R-CNN: Delving into High Quality Object Detection. CVPR 2018.

As mentioned in Cascade R-CNN, training with different IoU thresholds will lead to classifiers with corresponding quality.

Proposal Classification

- High quality object detection requires high IoU threshold, but directly raising the IoU threshold is impractical due to the vanishing positive samples [2]

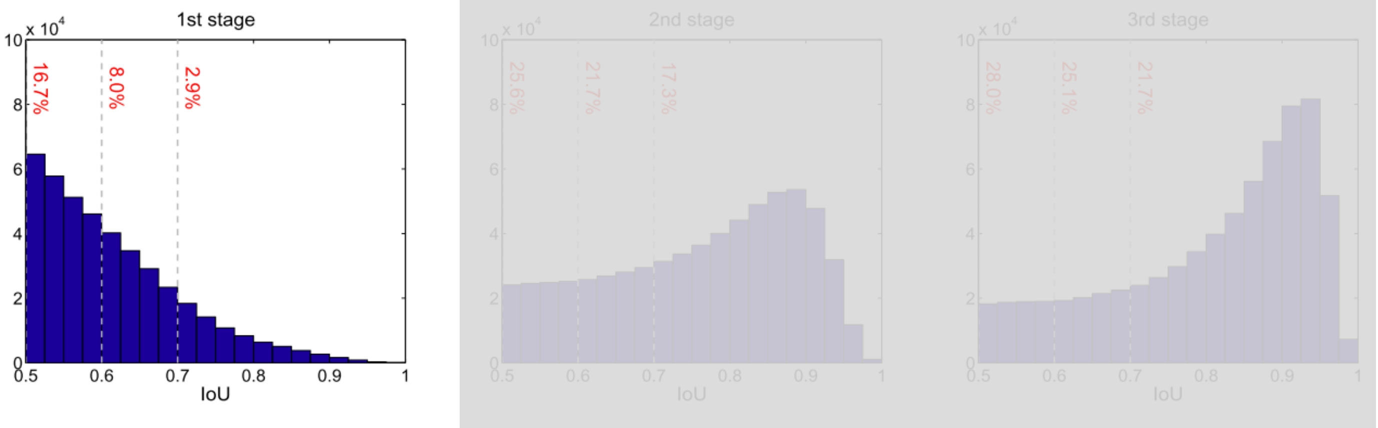


Fig. 4: IoU histograms of training samples of each cascade stage. The distribution of the 1st stage is the RPN output. Shown in red are the percentage of positives for the corresponding IoU threshold.

[2] Zhaowei Cai, et al. Cascade R-CNN: Delving into High Quality Object Detection. CVPR 2018.

To train a high quality classifier we need a high IoU threshold, but directly raising it will lead to overfitting due to the vanishing positives.

Proposal Classification

- Cascading several stages to lift the IoU of proposals [2]
- Effective yet time-consuming
- Better ideas?

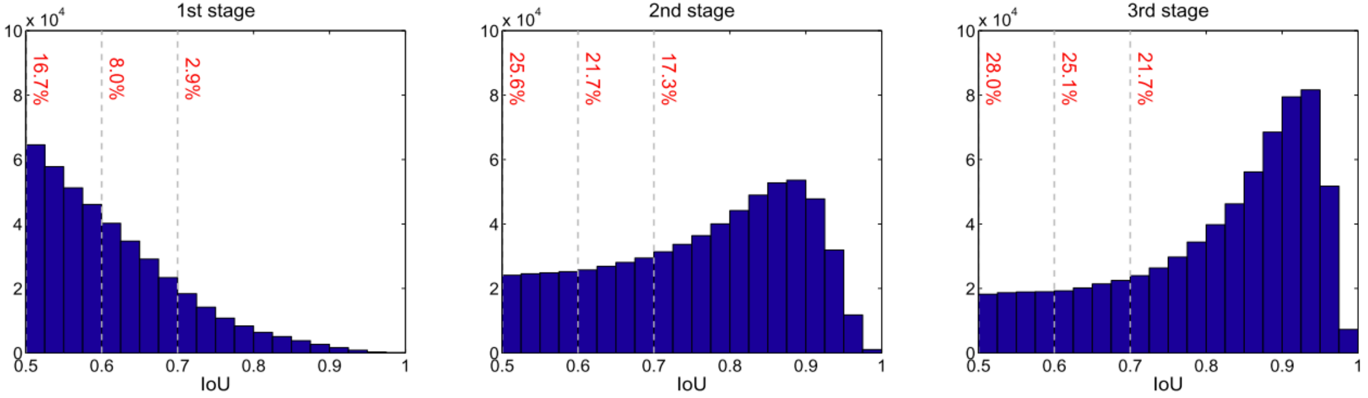


Fig. 4: IoU histograms of training samples of each cascade stage. The distribution of the 1st stage is the RPN output. Shown in red are the percentage of positives for the corresponding IoU threshold.

[2] Zhaowei Cai, et al. Cascade R-CNN: Delving into High Quality Object Detection. CVPR 2018.

So Cascade R-CNN adopts several sequential stages to lift the IoU of proposals, which is effective yet time-consuming. So are there any better ideas?

Proposal Classification

- The quality of proposals actually improves along the training process
- It inspires us to take a progressive approach in training

As mentioned before, the proposal quality actually improves over training. It inspires us to take a progressive approach.

Proposal Classification

- The quality of proposals actually improves along the training process
- It inspires us to take a progressive approach in training
    - At the beginning -> no enough high quality proposals -> low IoU threshold

At the beginning, RPN cannot produce enough high quality proposals, so we use a lower IoU threshold to better accommodate these imperfect proposals.

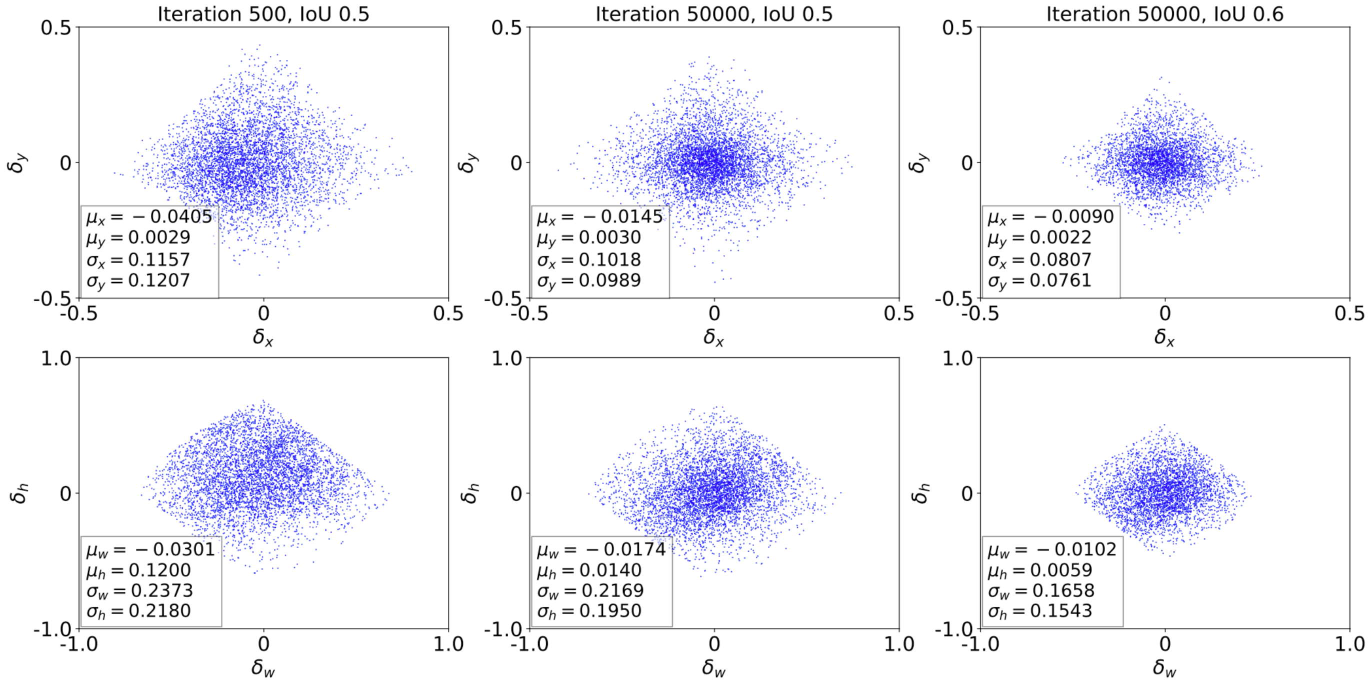
Proposal Classification

- The quality of proposals actually improves along the training process
- It inspires us to take a progressive approach in training
  - At the beginning -> no enough high quality proposals -> low IoU threshold
  - As the training goes -> proposal quality improves -> adjust IoU threshold

As training goes, the proposal quality improves and we gradually have enough high quality proposals. So we can raise the IoU threshold to train a high quality classifier.

Bounding Box Regression

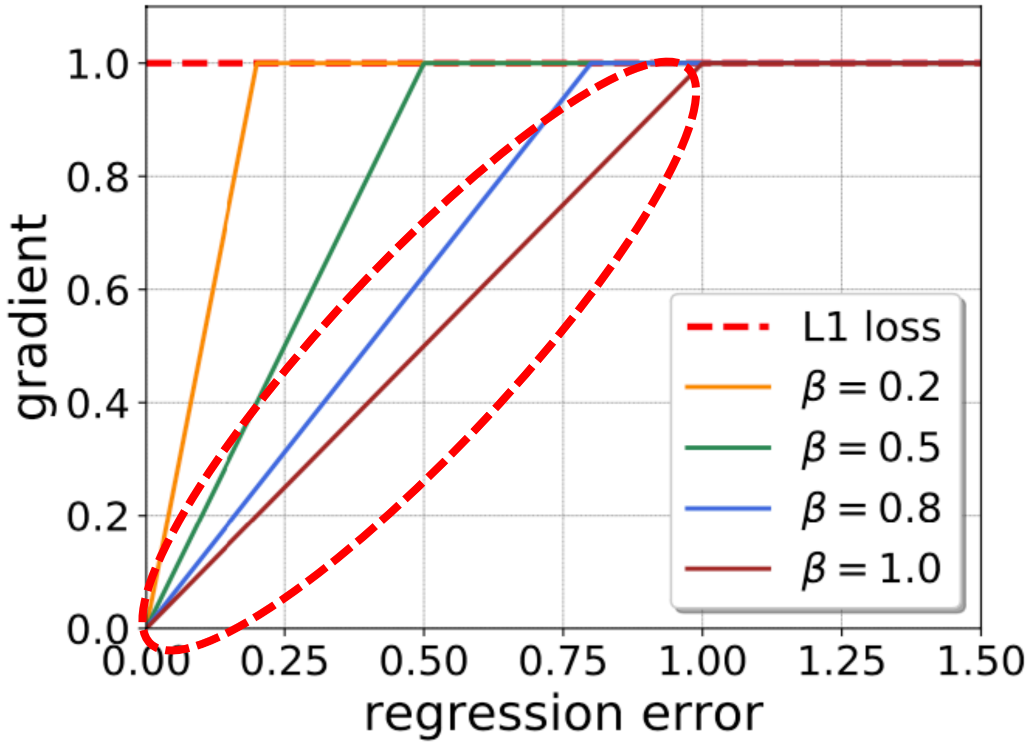
- Regression labels are shifting during training, showing the improved proposal quality



As for the regression task, we also find that the regression labels are shifting during training, showing the improved proposal quality.

Bounding Box Regression

- SmoothL1 Loss (default beta=1.0) will reduce the contributions of high quality samples (red circle)



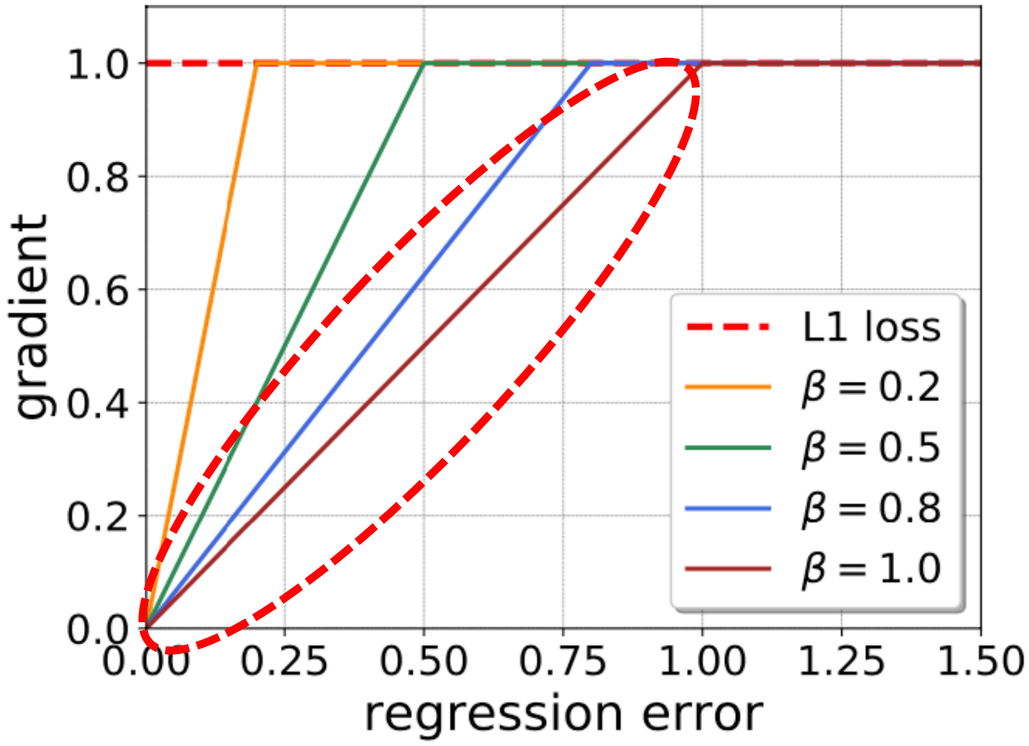
$$SmoothL1(x, \beta) = \begin{cases} 0.5|x|^2/\beta, & \text{if } |x| < \beta, \\ |x| - 0.5\beta, & \text{otherwise.} \end{cases}$$

However, the regression loss function will reduce the contributions of high quality samples, which is harmful to training high quality regressors.

Bounding Box Regression

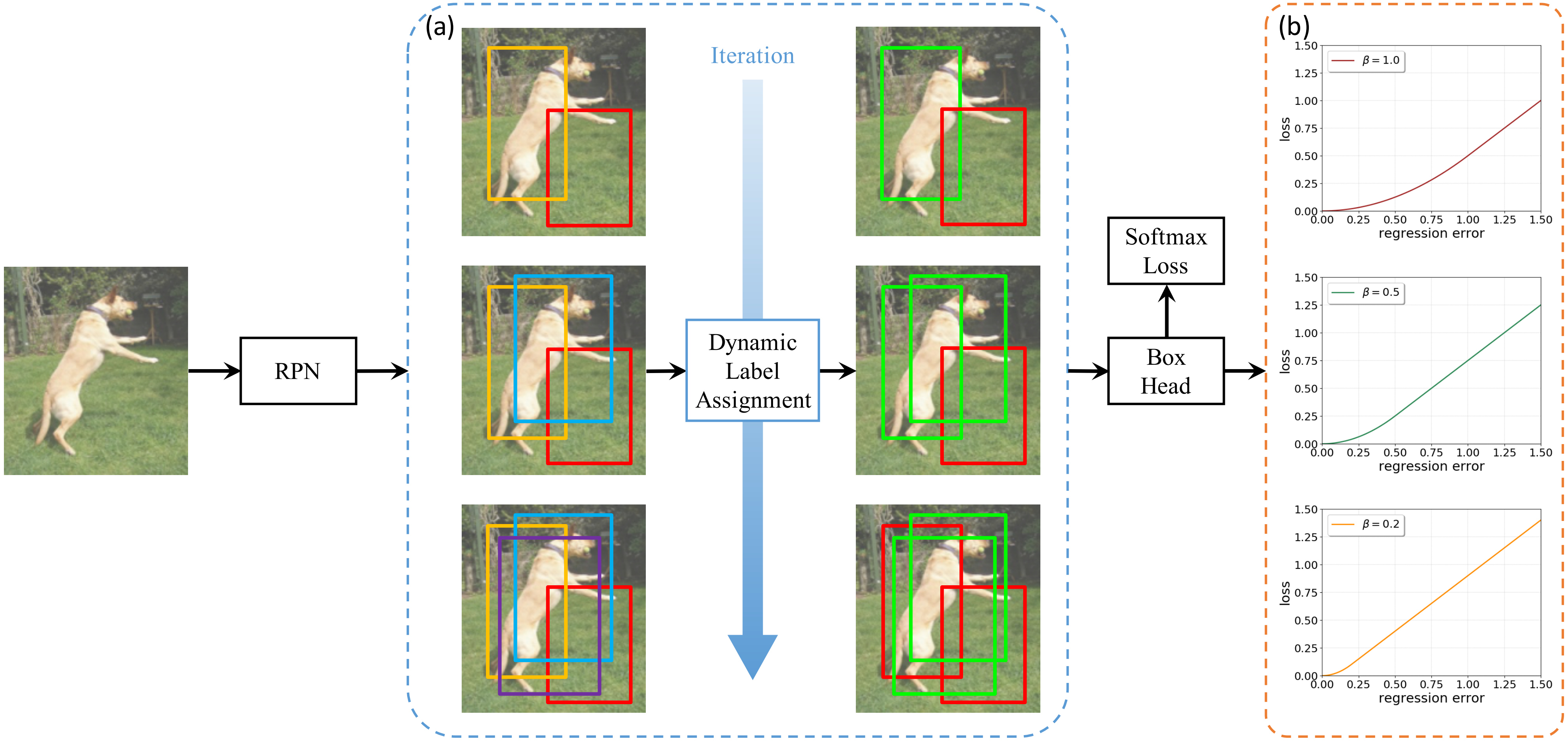
- SmoothL1 Loss (default beta=1.0) will reduce the contributions of high quality samples (red circle)
- Compensate for high quality samples

$$SmoothL1(x,\beta) = \begin{cases} 0.5|x|^2/\beta, & \text{if } |x| < \beta, \\ |x| - 0.5\beta, & \text{otherwise.} \end{cases}$$



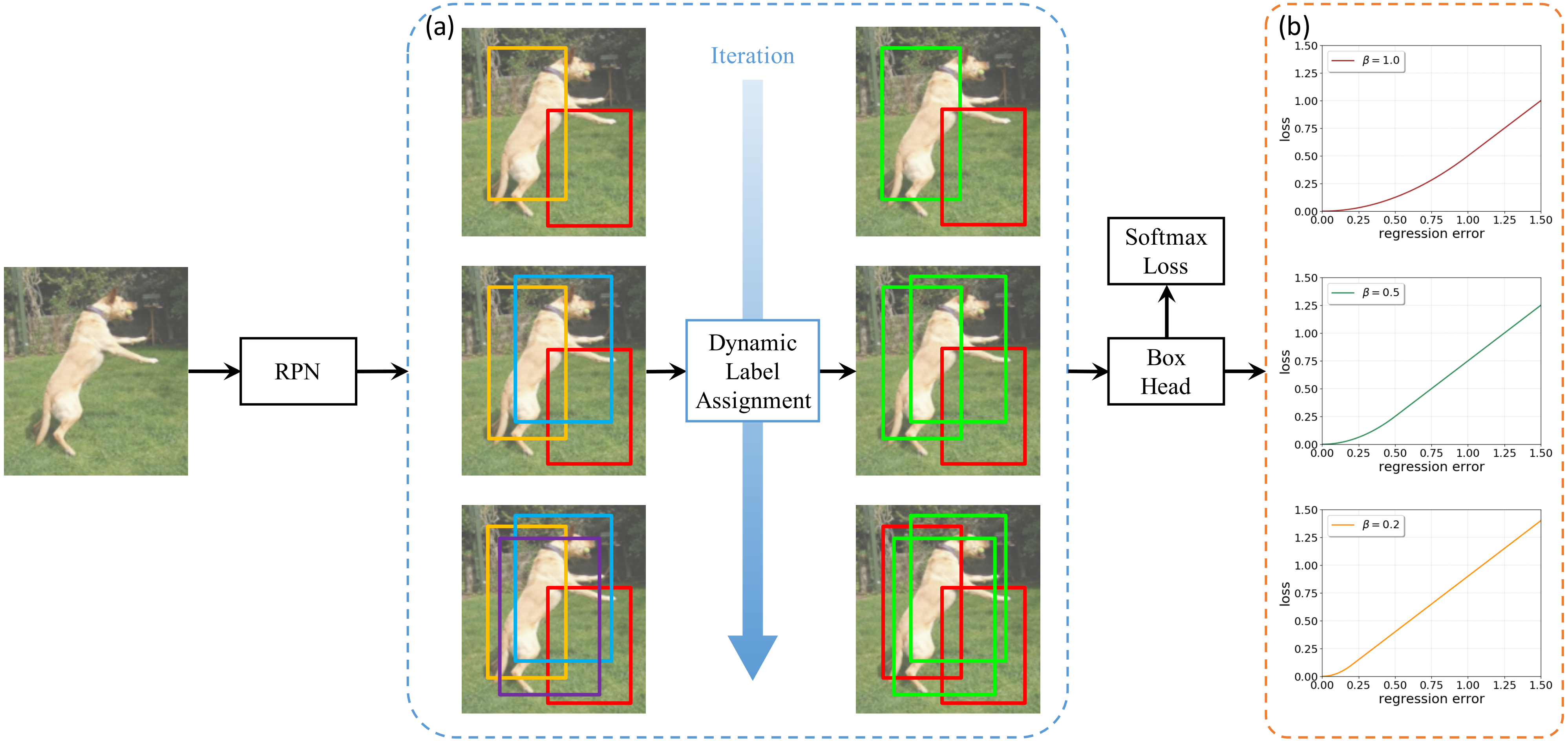
To achieve high quality object detection, we need to compensate for the high quality samples by adjusting the loss shape.

Dynamic R-CNN



To better exploit the dynamic property of the training procedure, we propose Dynamic R-CNN which consists of the following two components.

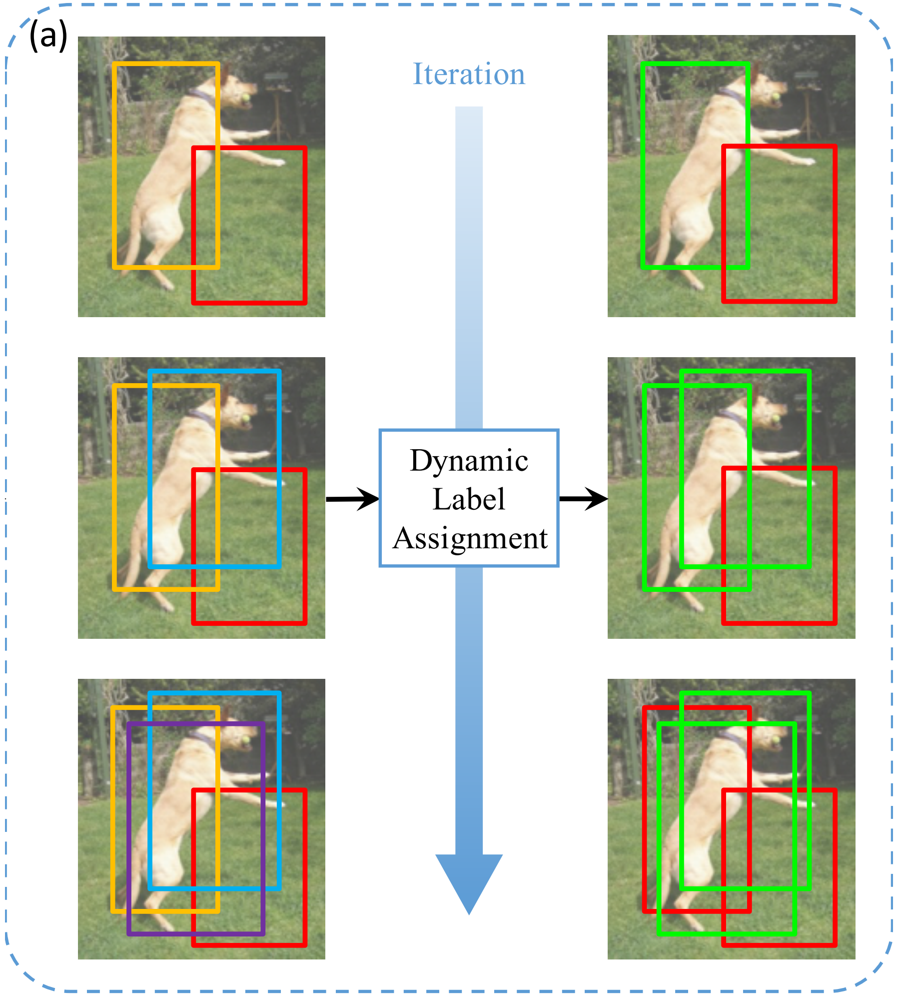
Dynamic R-CNN



Our key insight is adjusting the second stage classifier and regressor to fit the distribution change of proposals.

Dynamic Label Assignment (DLA) for Proposal Classification

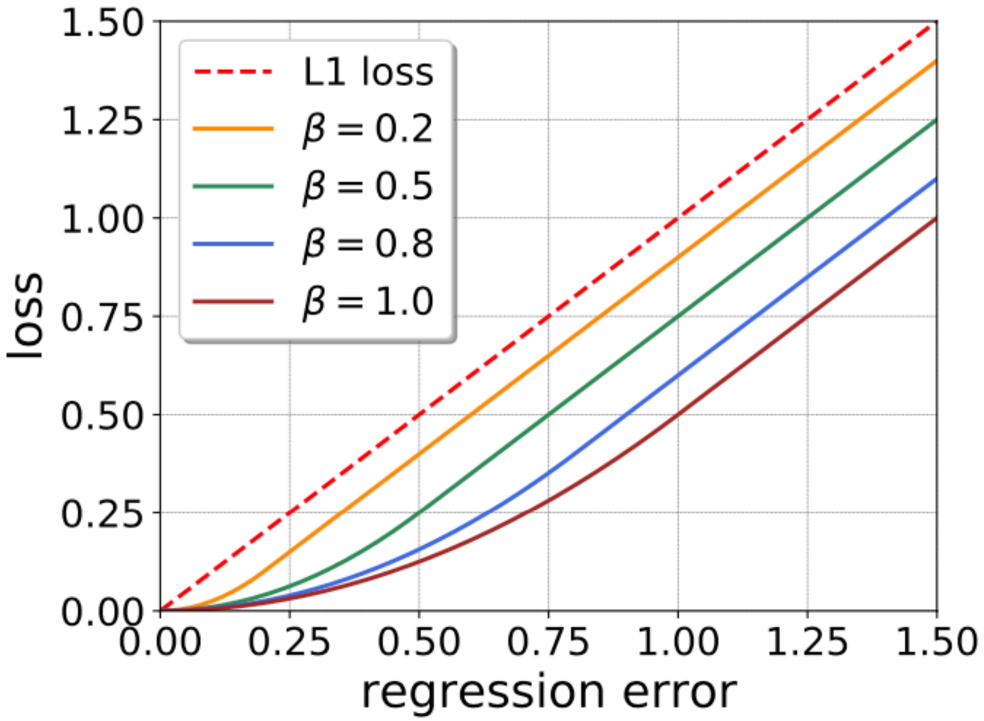
- DLA
  - Update the training IoU threshold according to the statistics of proposals
  - Specifically, we use the KI-th most accurate proposal's IoU to update the training IoU threshold



As for proposal classification, DLA can automatically update the IoU threshold based on proposal statistics, especially the IoU of the KI-th most accurate proposal.

Dynamic SmoothL1 Loss (DSL) for Bounding Box Regression

- DSL
  - DSL will update the beta of SmoothL1 Loss according to the statistics of proposals
  - Specifically, we use the Kbeta-th most accurate proposal's regression label to update the SmoothL1 beta, then the shape of loss function will changed



Dynamic R-CNN Algorithm

Algorithm 1 Dynamic R-CNN

Input:

Proposal set P, ground-truth set G.

IoU threshold top-k K_I, β top-k K_β, update iteration count C.

Output:

Trained object detector D.

```
Initialize IoU threshold and SmoothL1 β as T_now, β_now
Build two empty sets S_I, S_E for recording the IoUs and regression labels
for i = 0 to max_iter do
    Obtain matched IoUs I and regression labels E between P and G
    Select thresholds I_k, E_k based on the K_I, K_β
    Record corresponding values, add I_k to S_I and E_k to S_E
    if i % C == 0 then
        Update IoU threshold: T_now = Mean(S_I)
        Update SmoothL1 β: β_now = Median(S_E)
        S_I = Ø, S_E = Ø
    Train the network with T_now, β_now
return Improved object detector D
```

To summarize the whole method, we describe Dynamic R-CNN in Algorithm 1.

Dynamic R-CNN Algorithm

Algorithm 1 Dynamic R-CNN

Input:

Proposal set P, ground-truth set G.

IoU threshold top-k K_I, β top-k K_β, update iteration count C.

Output:

Trained object detector D.

Initialize IoU threshold and SmoothL1 β as T_{now}, β_{now}
Build two empty sets $\mathcal{S}_I, \mathcal{S}_E$ for recording the IoUs and regression labels
for $i = 0$ to max_iter **do**
    Obtain matched IoUs I and regression labels E between P and G
    Select thresholds I_k, E_k based on the K_I, K_β
    Record corresponding values, add I_k to $\mathcal{S}_I$ and E_k to $\mathcal{S}_E$
    if $i \,\%\, C == 0$ **then**
        Update IoU threshold: $T_{now} = \text{Mean}(\mathcal{S}_I)$
        Update SmoothL1 β: $\beta_{now} = \text{Median}(\mathcal{S}_E)$
        $\mathcal{S}_I = \emptyset, \mathcal{S}_E = \emptyset$
    Train the network with T_{now}, β_{now}
return Improved object detector D

First, for each iteration, Dynamic R-CNN obtains the matched IoUs and regression labels, then select and record the corresponding statistics.

Dynamic R-CNN Algorithm

Algorithm 1 Dynamic R-CNN

Input:

Proposal set P, ground-truth set G.
IoU threshold top-k K_I, β top-k K_β, update iteration count C.

Output:

Trained object detector D.

Initialize IoU threshold and SmoothL1 β as T_{now}, β_{now}
Build two empty sets $\mathcal{S}_I, \mathcal{S}_E$ for recording the IoUs and regression labels
for $i = 0$ to max_iter **do**
    Obtain matched IoUs I and regression labels E between P and G
    Select thresholds I_k, E_k based on the K_I, K_β
    Record corresponding values, add I_k to $\mathcal{S}_I$ and E_k to $\mathcal{S}_E$
    if $i \,\%\, C == 0$ **then**
        Update IoU threshold: $T_{now} = \text{Mean}(\mathcal{S}_I)$
        Update SmoothL1 β: $\beta_{now} = \text{Median}(\mathcal{S}_E)$
        $\mathcal{S}_I = \emptyset, \mathcal{S}_E = \emptyset$
    Train the network with T_{now}, β_{now}
return Improved object detector D

It should be noted that the matched IoUs and regression labels are already obtained by the original method, so we introduce almost no extra overhead.

Dynamic R-CNN Algorithm

Algorithm 1 Dynamic R-CNN

Input:

Proposal set P, ground-truth set G.

IoU threshold top-k K_I, β top-k K_β, update iteration count C.

Output:

Trained object detector D.

Initialize IoU threshold and SmoothL1 β as T_{now}, β_{now}
Build two empty sets $\mathcal{S}_I, \mathcal{S}_E$ for recording the IoUs and regression labels
for $i = 0$ to max_iter **do**
  Obtain matched IoUs I and regression labels E between P and G
  Select thresholds I_k, E_k based on the K_I, K_β
  Record corresponding values, add I_k to $\mathcal{S}_I$ and E_k to $\mathcal{S}_E$
  if $i \% C == 0$ **then**
    Update IoU threshold: $T_{now} = \mathrm{Mean}(\mathcal{S}_I)$
    Update SmoothL1 β: $\beta_{now} = \mathrm{Median}(\mathcal{S}_E)$
    $\mathcal{S}_I = \emptyset, \mathcal{S}_E = \emptyset$
  Train the network with T_{now}, β_{now}
return Improved object detector D

Then, Dynamic R-CNN will update the IoU threshold and SmoothL1 beta every C iterations to fit the dynamic property during training.

Dynamic R-CNN Algorithm

Algorithm 1 Dynamic R-CNN

Input:

Proposal set P, ground-truth set G.

IoU threshold top-k K_I, β top-k K_β, update iteration count C.

Output:

Trained object detector D.

Initialize IoU threshold and SmoothL1 β as T_{now}, β_{now}
Build two empty sets $\mathcal{S}_I, \mathcal{S}_E$ for recording the IoUs and regression labels
for $i = 0$ to max_iter **do**
    Obtain matched IoUs I and regression labels E between P and G
    Select thresholds I_k, E_k based on the K_I, K_β
    Record corresponding values, add I_k to $\mathcal{S}_I$ and E_k to $\mathcal{S}_E$
    if $i \ \% \ C == 0$ **then**
        Update IoU threshold: $T_{now} = \text{Mean}(\mathcal{S}_I)$
        Update SmoothL1 β: $\beta_{now} = \text{Median}(\mathcal{S}_E)$
        $\mathcal{S}_I = \emptyset, \mathcal{S}_E = \emptyset$
    Train the network with T_{now}, β_{now}
$\boxed{\textbf{return} \text{ Improved object detector } D}$

Finally, we get an improved object detector.

Experiments (Main Result)

Table 1. Comparisons with different baselines (our re-implementations) on COCO `test-dev` set. "MST" and "*" stand for multi-scale training and testing respectively. "2×" and "3×" are training schedules which extend the iterations by 2/3 times.

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN	ResNet-50	37.3	58.5	40.6	20.3	39.2	49.1
Faster R-CNN+2×	ResNet-50	38.1	58.9	41.5	20.5	40.0	50.0
Faster R-CNN	ResNet-101	39.3	60.5	42.7	21.3	41.8	51.7
Faster R-CNN+2×	ResNet-101	39.9	60.6	43.5	21.4	42.4	52.1
Faster R-CNN+3×+MST	ResNet-101	42.8	63.8	46.8	24.8	45.6	55.6
Faster R-CNN+3×+MST	ResNet-101-DCN	44.8	65.5	48.8	26.2	47.6	58.1
Faster R-CNN+3×+MST*	ResNet-101-DCN	46.9	68.1	51.4	30.6	49.6	58.1
Dynamic R-CNN	ResNet-50	39.1	58.0	42.8	21.3	40.9	50.3
Dynamic R-CNN+2×	ResNet-50	39.9	58.6	43.7	21.6	41.5	51.9
Dynamic R-CNN	ResNet-101	41.2	60.1	45.1	22.5	43.6	53.2
Dynamic R-CNN+2×	ResNet-101	42.0	60.7	45.9	22.7	44.3	54.3
Dynamic R-CNN+3×+MST	ResNet-101	44.7	63.6	49.1	26.0	47.4	57.2
Dynamic R-CNN+3×+MST	ResNet-101-DCN	46.9	65.9	51.3	28.1	49.6	60.0
Dynamic R-CNN+3×+MST*	ResNet-101-DCN	49.2	68.6	54.0	32.5	51.7	60.3

Detailed experiments are provided to show our effectiveness. First, we compare Dynamic R-CNN with corresponding baselines under different settings.

Experiments (Main Result)

Table 1. Comparisons with different baselines (our re-implementations) on COCO `test-dev` set. "MST" and "*" stand for multi-scale training and testing respectively. "2×" and "3×" are training schedules which extend the iterations by 2/3 times.

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN	ResNet-50	37.3	58.5	40.6	20.3	39.2	49.1
Faster R-CNN+2×	ResNet-50	38.1	58.9	41.5	20.5	40.0	50.0
Faster R-CNN	ResNet-101	39.3	60.5	42.7	21.3	41.8	51.7
Faster R-CNN+2×	ResNet-101	39.9	60.6	43.5	21.4	42.4	52.1
Faster R-CNN+3×+MST	ResNet-101	42.8	63.8	46.8	24.8	45.6	55.6
Faster R-CNN+3×+MST	ResNet-101-DCN	44.8	65.5	48.8	26.2	47.6	58.1
Faster R-CNN+3×+MST*	ResNet-101-DCN	46.9	68.1	51.4	30.6	49.6	58.1
Dynamic R-CNN	ResNet-50	39.1	58.0	42.8	21.3	40.9	50.3
Dynamic R-CNN+2×	ResNet-50	39.9	58.6	43.7	21.6	41.5	51.9
Dynamic R-CNN	ResNet-101	41.2	60.1	45.1	22.5	43.6	53.2
Dynamic R-CNN+2×	ResNet-101	42.0	60.7	45.9	22.7	44.3	54.3
Dynamic R-CNN+3×+MST	ResNet-101	44.7	63.6	49.1	26.0	47.4	57.2
Dynamic R-CNN+3×+MST	ResNet-101-DCN	46.9	65.9	51.3	28.1	49.6	60.0
Dynamic R-CNN+3×+MST*	ResNet-101-DCN	49.2	68.6	54.0	32.5	51.7	60.3

As shown in Table 1, our method can work on different backbones and it is also compatible with other training and testing skills.

Experiments (Main Result)

Table 1. Comparisons with different baselines (our re-implementations) on COCO `test-dev` set. "MST" and "*" stand for multi-scale training and testing respectively. "2×" and "3×" are training schedules which extend the iterations by 2/3 times.

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN	ResNet-50	37.3	58.5	40.6	20.3	39.2	49.1
Faster R-CNN+2×	ResNet-50	38.1	58.9	41.5	20.5	40.0	50.0
Faster R-CNN	ResNet-101	39.3	60.5	42.7	21.3	41.8	51.7
Faster R-CNN+2×	ResNet-101	39.9	60.6	43.5	21.4	42.4	52.1
Faster R-CNN+3×+MST	ResNet-101	42.8	63.8	46.8	24.8	45.6	55.6
Faster R-CNN+3×+MST	ResNet-101-DCN	44.8	65.5	48.8	26.2	47.6	58.1
Faster R-CNN+3×+MST*	ResNet-101-DCN	46.9	68.1	51.4	30.6	49.6	58.1
Dynamic R-CNN	ResNet-50	39.1	58.0	42.8	21.3	40.9	50.3
Dynamic R-CNN+2×	ResNet-50	39.9	58.6	43.7	21.6	41.5	51.9
Dynamic R-CNN	ResNet-101	41.2	60.1	45.1	22.5	43.6	53.2
Dynamic R-CNN+2×	ResNet-101	42.0	60.7	45.9	22.7	44.3	54.3
Dynamic R-CNN+3×+MST	ResNet-101	44.7	63.6	49.1	26.0	47.4	57.2
Dynamic R-CNN+3×+MST	ResNet-101-DCN	46.9	65.9	51.3	28.1	49.6	60.0
Dynamic R-CNN+3×+MST*	ResNet-101-DCN	49.2	68.6	54.0	32.5	51.7	60.3

Generally speaking, our method can improve different baselines by almost 2 points AP consistently.

Experiments (Components)

Table 2. Results of each component in Dynamic R-CNN on COCO `val` set.

Backbone	DLA	DSL	AP	ΔAP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
ResNet-50-FPN			37.0	-	58.0	53.5	46.0	32.6	9.7
ResNet-50-FPN		✓	38.0	+1.0	57.6	53.5	46.7	34.4	13.2
ResNet-50-FPN	✓		38.2	+1.2	57.5	53.6	47.1	35.2	12.6
ResNet-50-FPN	✓	✓	38.9	+1.9	57.3	53.6	47.4	36.3	15.2

Then, we conduct experiments to validate the effectiveness of different components in Dynamic R-CNN.

Experiments (Components)

Table 2. Results of each component in Dynamic R-CNN on COCO `val` set.

Backbone	DLA	DSL	AP	ΔAP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
ResNet-50-FPN			37.0	-	58.0	53.5	46.0	32.6	9.7
ResNet-50-FPN		✓	38.0	+1.0	57.6	53.5	46.7	34.4	13.2
ResNet-50-FPN	✓		38.2	+1.2	57.5	53.6	47.1	35.2	12.6
ResNet-50-FPN	✓	✓	38.9	+1.9	57.3	53.6	47.4	36.3	15.2

Compared to the Faster R-CNN baseline, DLA and DSL can bring 1.2 and 1.0 points higher box AP, and the improvements mainly lie in higher IoU metrics.

Experiments (Components)

Table 2. Results of each component in Dynamic R-CNN on COCO `val` set.

Backbone	DLA	DSL	AP	ΔAP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
ResNet-50-FPN			37.0	-	58.0	53.5	46.0	32.6	9.7
ResNet-50-FPN		✓	38.0	+1.0	57.6	53.5	46.7	34.4	13.2
ResNet-50-FPN	✓		38.2	+1.2	57.5	53.6	47.1	35.2	12.6
ResNet-50-FPN	✓	✓	38.9	+1.9	57.3	53.6	47.4	36.3	15.2

Since with DLA the contributions of positives are reduced even more, DLA and DSL can work together.

Experiments (Components)

Table 2. Results of each component in Dynamic R-CNN on COCO `val` set.

Backbone	DLA	DSL	AP	ΔAP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
ResNet-50-FPN			37.0	-	58.0	53.5	46.0	32.6	9.7
ResNet-50-FPN		✓	38.0	+1.0	57.6	53.5	46.7	34.4	13.2
ResNet-50-FPN	✓		38.2	+1.2	57.5	53.6	47.1	35.2	12.6
ResNet-50-FPN	✓	✓	38.9	+1.9	57.3	53.6	47.4	36.3	15.2

+5.5 AP90

To sum up, Dynamic R-CNN improves the baseline by 1.9 points AP and 5.5 points AP90.

Experiments (Dynamic trends)

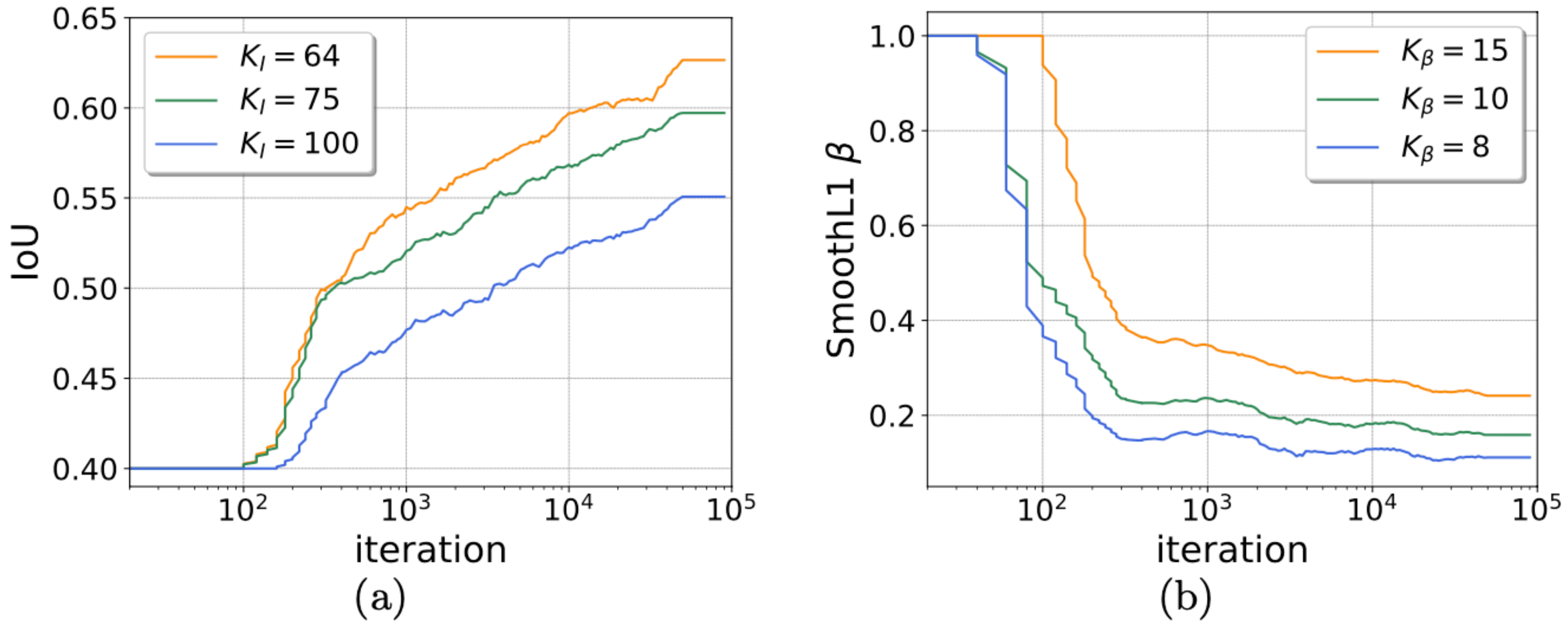


Fig. 5. Trends of (a) IoU threshold and (b) SmoothL1 β under different settings based on our method. Obviously the distribution has changed a lot during training.

To further illustrate the dynamics in training, we show the trends of IoU threshold and SmoothL1 beta under different settings in Figure 5.

Experiments (Dynamic trends)

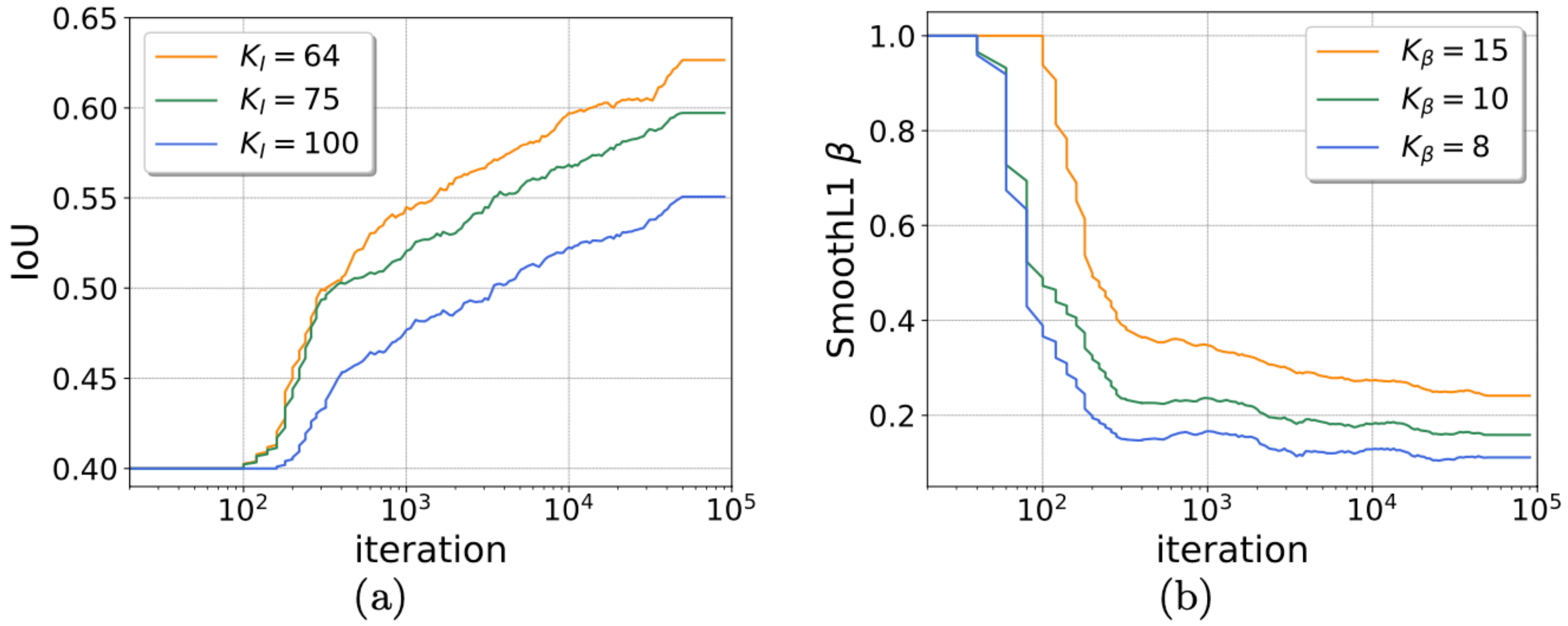


Fig. 5. Trends of (a) IoU threshold and (b) SmoothL1 β under different settings based on our method. Obviously the distribution has changed a lot during training.

Regardless of the specific settings, the trend of IoU threshold is increasing while that for beta is decreasing during training as expected.

Experiments (Hyper-parameters)

Table 3. Ablation study on K_I.

K_I	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
-	37.0	58.0	53.5	46.0	32.6	9.7
64	38.1	57.2	53.3	46.8	35.1	**12.8**
75	**38.2**	57.5	53.6	**47.1**	**35.2**	12.6
100	37.9	**57.9**	**53.8**	46.9	34.2	11.6

Table 4. Ablation study on C.

C	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
-	37.0	58.0	53.5	46.0	32.6	9.7
20	38.0	57.4	53.5	47.0	35.0	12.5
100	38.2	57.5	53.6	47.1	35.2	12.6
500	38.1	57.6	53.5	47.2	34.8	12.6

Table 5. Ablation study on K_β.

Setting	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
$\beta = 1.0$	37.0	58.0	53.5	46.0	32.6	9.7
$\beta = 2.0$	35.9	57.7	53.2	45.1	30.1	8.3
$\beta = 0.5$	37.5	57.6	53.3	46.4	33.5	11.3
$K_\beta = 15$	37.6	57.3	53.1	46.0	33.9	12.5
$K_\beta = 10$	38.0	57.6	53.5	46.7	34.4	13.2
$K_\beta = 8$	37.6	57.5	53.3	45.9	33.9	12.4

Then, experiments on the effect of hyper-parameters are shown in these tables. Generally speaking, the results are not sensitive to these hyper-parameters.

Experiments (Hyper-parameters)

Table 3. Ablation study on K_I.

K_I	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
-	37.0	58.0	53.5	46.0	32.6	9.7
64	38.1	57.2	53.3	46.8	35.1	**12.8**
75	**38.2**	57.5	53.6	**47.1**	**35.2**	12.6
100	37.9	**57.9**	**53.8**	46.9	34.2	11.6

Table 5. Ablation study on K_β.

Setting	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
$\beta = 1.0$	37.0	58.0	53.5	46.0	32.6	9.7
$\beta = 2.0$	35.9	57.7	53.2	45.1	30.1	8.3
$\beta = 0.5$	37.5	57.6	53.3	46.4	33.5	11.3
$K_\beta = 15$	37.6	57.3	53.1	46.0	33.9	12.5
$K_\beta = 10$	38.0	57.6	53.5	46.7	34.4	13.2
$K_\beta = 8$	37.6	57.5	53.3	45.9	33.9	12.4

As for DLA and DSL, choosing different KI or Kbeta will lead to different performances on different metrics. But the final box AP is similar.

Experiments (Hyper-parameters)

Table 4. Ablation study on C.

C	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
-	37.0	58.0	53.5	46.0	32.6	9.7
20	38.0	57.4	53.5	47.0	35.0	12.5
100	38.2	57.5	53.6	47.1	35.2	12.6
500	38.1	57.6	53.5	47.2	34.8	12.6

And the result is also robust to the value of iteration count C.

Experiments (Hyper-parameters)

- Only one additional hyper-parameter
  - Faster R-CNN IoU threshold -> KI
  - Faster R-CNN SmoothL1 beta -> Kbeta
  - Iteration count C (additional but robust)

It should be noted that compared to the IoU threshold and beta in Faster R-CNN, we only add one hyper-parameter C and the results are robust to it.

Experiments (Complexity and Speed)

- Comparison to High Quality detectors (e.g. Cascade R-CNN)
  - Advantages
    - Does not increase the training time (Basically)

Our advantage compared to Faster R-CNN is the efficiency. On one hand, Dynamic R-CNN does not increase the training time.

Experiments (Complexity and Speed)

- Comparison to High Quality detectors (e.g. Cascade R-CNN)
  - Advantages
    - Does not increase the training time (Basically)
    - Faster inference speed

Table 6. Inference speed comparisons using ResNet-50-FPN backbone on RTX 2080TI GPU.

Method	FPS
Dynamic R-CNN	13.9
Cascade R-CNN	11.2
Dynamic Mask R-CNN	11.3
Cascade Mask R-CNN	7.3

On the other hand, the inference speed of Dynamic R-CNN is faster, and the speedup ratio depends on the specific backbone and the head design.

Experiments (Complexity and Speed)

- Comparison to High Quality detectors (e.g. Cascade R-CNN)
  - Advantages
    - Does not increase the training time (Basically)
    - Faster inference speed (1.74 times faster using ResNet-18 with mask head)

Table 6. Inference speed comparisons using ResNet-50-FPN backbone on RTX 2080TI GPU.

Method	FPS
Dynamic R-CNN	13.9
Cascade R-CNN	11.2
Dynamic Mask R-CNN	11.3
Cascade Mask R-CNN	7.3

For instance, using ResNet-18 as the backbone, Dynamic Mask R-CNN is 1.74 times faster than Cascade Mask R-CNN.

Experiments (Universality)

Table 7. The universality of Dynamic R-CNN. We apply the idea of dynamic training on Mask R-CNN under different backbones. "bbox" and "segm" stand for object detection and instance segmentation results on COCO `val` set, respectively.

Backbone	+Dynamic	AP^{bbox}	AP_{50}^{bbox}	AP_{75}^{bbox}	AP^{segm}	AP_{50}^{segm}	AP_{75}^{segm}
ResNet-50-FPN		37.5	58.0	40.7	33.8	54.6	36.0
	✓	39.4	57.6	43.3	34.8	55.0	37.5
ResNet-101-FPN		39.7	60.7	43.2	35.6	56.9	37.7
	✓	41.8	60.4	45.8	36.7	57.5	39.4

Moreover, since our dynamic viewpoint is a general concept, we believe it can be adopted in different methods.

Experiments (Universality)

Table 7. The universality of Dynamic R-CNN. We apply the idea of dynamic training on Mask R-CNN under different backbones. "bbox" and "segm" stand for object detection and instance segmentation results on COCO `val` set, respectively.

Backbone	+Dynamic	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{segm}	AP^{segm}_{50}	AP^{segm}_{75}
ResNet-50-FPN		37.5	58.0	40.7	33.8	54.6	36.0
	✓	39.4	57.6	43.3	34.8	55.0	37.5
ResNet-101-FPN		39.7	60.7	43.2	35.6	56.9	37.7
	✓	41.8	60.4	45.8	36.7	57.5	39.4

We apply the dynamic design on Mask R-CNN with different backbones and find that both the detection and instance segmentation results are improved.

Experiments (Universality)

Table 7. The universality of Dynamic R-CNN. We apply the idea of dynamic training on Mask R-CNN under different backbones. "bbox" and "segm" stand for object detection and instance segmentation results on COCO `val` set, respectively.

Backbone	+Dynamic	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{segm}	AP^{segm}_{50}	AP^{segm}_{75}
ResNet-50-FPN		37.5	58.0	40.7	33.8	54.6	36.0
	✓	39.4	57.6	43.3	34.8	55.0	37.5
ResNet-101-FPN		39.7	60.7	43.2	35.6	56.9	37.7
	✓	41.8	60.4	45.8	36.7	57.5	39.4

Note that we only adopt the DLA and DSL which are designed for object detection, so the results further demonstrate our effectiveness and universality.

Experiments (State-of-the-Arts)

Table 8. Comparisons of single-model results on COCO `test-dev` set.

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet [28]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
CornerNet [23]	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
FCOS [42]	ResNet-101	41.0	60.7	44.1	24.0	44.1	51.0
FreeAnchor [49]	ResNet-101	41.8	61.1	44.9	22.6	44.7	53.9
RepPoints [46]	ResNet-101-DCN	45.0	66.1	49.0	26.6	48.6	57.5
CenterNet [50]	Hourglass-104	45.1	63.9	49.3	26.6	47.1	57.7
ATSS [48]	ResNet-101-DCN	46.3	64.7	50.4	27.7	49.8	58.4
Faster R-CNN [27]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2
Mask R-CNN [14]	ResNet-101	38.2	60.3	41.7	20.1	41.1	50.2
Regionlets [45]	ResNet-101	39.3	59.8	-	21.7	43.7	50.9
Libra R-CNN [33]	ResNet-101	41.1	62.1	44.7	23.4	43.7	52.5
Cascade R-CNN [3]	ResNet-101	42.8	62.1	46.3	23.7	45.5	55.2
SNIP [40]	ResNet-101-DCN	44.4	66.2	49.9	27.3	47.4	56.9
DCNv2 [51]	ResNet-101-DCN	46.0	67.9	50.8	27.8	49.1	59.5
TridentNet [25]	ResNet-101-DCN	48.4	69.7	53.5	31.8	51.3	60.3
Dynamic R-CNN	ResNet-101	42.0	60.7	45.9	22.7	44.3	54.3
Dynamic R-CNN*	ResNet-101-DCN	50.1	68.3	55.6	32.8	53.0	61.2

Finally, we compare Dynamic R-CNN with the state-of-the-arts and find that our method outperforms other previous detectors.

Conclusion

- Dynamic R-CNN can bring consistent gains with no extra overhead, which is a free lunch for high quality object detection ~

In Conclusion, our method brings consistent gains with no extra overhead, which is a free lunch for high quality object detection.

Conclusion

- Dynamic R-CNN can bring consistent gains with no extra overhead, which is a free lunch for high quality object detection ~
- We hope that this dynamic viewpoint can inspire further researches in the future

Moreover, we hope our dynamic viewpoint can inspire further researches in the future.

ECCV'20 ONLINE
23-28 AUGUST 2020

Thank you!

Kevin.hkzhang@gmail.com

Source code